

Package ‘EnrichmentBrowser’

April 15, 2019

Version 2.12.1

Date 2018-10-18

Title Seamless navigation through combined results of set-based and network-based enrichment analysis

Author Ludwig Geistlinger [aut, cre], Gergely Csaba [aut], Mara Santarelli [ctb], Marcel Ramos [ctb], Levi Waldron [ctb], Ralf Zimmer [aut]

Maintainer Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

Depends R(>= 3.5.0), SummarizedExperiment, graph

Imports AnnotationDbi, BiocFileCache, BiocManager, ComplexHeatmap, DESeq2, EDASeq, GSEABase, GO.db, KEGGREST, KEGGgraph, MASS, ReportingTools, Rgraphviz, S4Vectors, SPIA, biocGraph, edgeR, geneplotter, graphite, hwriter, limma, methods, pathview, rappdirs, safe, topGO

Suggests ALL, BiocStyle, airway, hgu95av2.db, knitr

Description The EnrichmentBrowser package implements essential functionality for the enrichment analysis of gene expression data. The analysis combines the advantages of set-based and network-based enrichment analysis in order to derive high-confidence gene sets and biological pathways that are differentially regulated in the expression data under investigation. Besides, the package facilitates the visualization and exploration of such sets and pathways.

License Artistic-2.0

Encoding UTF-8

VignetteBuilder knitr

biocViews ImmunoOncology, Microarray, RNASeq, GeneExpression, DifferentialExpression, Pathways, GraphAndNetwork, Network, GeneSetEnrichment, NetworkEnrichment, Visualization, ReportWriting

RoxygenNote 6.1.0

git_url <https://git.bioconductor.org/packages/EnrichmentBrowser>

git_branch RELEASE_3_8

git_last_commit 9b180b3

git_last_commit_date 2019-01-04

Date/Publication 2019-04-15

R topics documented:

combResults	2
compileGRN	4
configEBrowser	5
deAna	7
downloadPathways	8
eaBrowse	9
ebrowser	10
getGenesets	14
ggeaGraph	16
idMap	17
isAvailable	19
makeExampleData	20
nbeaMethods	21
normalize	25
plots	26
probe2gene	27
readSE	29
sbeaMethods	30
Index	34

combResults

Combining enrichment analysis results

Description

Different enrichment analysis methods usually result in different gene set rankings for the same dataset. This function allows to combine results from the different set-based and network-based enrichment analysis methods. This includes the computation of average gene set ranks across methods.

Usage

```
combResults(res.list, rank.col = configEBrowser("PVAL.COL"),
  decreasing = FALSE, rank.fun = c("comp.ranks", "rel.ranks",
  "abs.ranks"), comb.fun = c("mean", "median", "min", "max", "sum"))
```

Arguments

- | | |
|------------|---|
| res.list | A list of enrichment analysis result lists (as returned by the functions sbea and nbea). |
| rank.col | Rank column. Column name of the enrichment analysis result table that should be used to rank the gene sets. Defaults to the gene set p-value column, i.e. gene sets are ranked according to gene set significance. |
| decreasing | Logical. Should smaller (decreasing=FALSE, default) or larger (decreasing=TRUE) values in rank.col be ranked better? In case of gene set p-values the smaller the better, in case of gene set scores the larger the better. |

rank.fun	<p>Ranking function. Used to rank gene sets according to the result table of individual enrichment methods (as returned from the gsRanking function). This is typically done according to gene set p-values, but can also take into account gene set scores/statistics, especially in case of gene sets with equal p-value. Can be either one of the predefined functions ('comp.ranks', 'rel.ranks', 'abs.ranks') or a user-defined function. Defaults to 'comp.ranks', i.e. competitive (percentile) ranks are computed by calculating for each gene set the percentage of gene sets with a p-value as small or smaller. Alternatively, 'rel.ranks', i.e. relative ranks are computed in 2 steps:</p> <ol style="list-style-type: none">1. Ranks are assigned according to distinct gene set p-value <i>categories</i>, i.e. gene sets with equal p-value obtain the <i>same</i> rank. Thus, the gene sets with lowest p-value obtain rank 1, and so on.2. As opposed to absolute ranks (rank.fun = 'abs.ranks'), which are returned from step 1, relative ranks are then computed by dividing the absolute rank by number of distinct p-value categories and multiplying with 100 (= percentile rank).
comb.fun	<p>Rank combination function. Used to combine gene set ranks across methods. Can be either one of the predefined functions (mean, median, max, min, sum) or a user-defined function. Defaults to 'sum', i.e. the rank sum across methods is computed.</p>

Value

An enrichment analysis result list that can be detailedly explored by calling [eaBrowse](#) and from which a flat gene set ranking can be extracted by calling [gsRanking](#).

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

See Also

[sbea](#), [nbea](#), [eaBrowse](#)

Examples

```
# (1) expression data:
# simulated expression values of 100 genes
# in two sample groups of 6 samples each
se <- makeExampleData(what="SE")
se <- deAna(se)

# (2) gene sets:
# draw 10 gene sets with 15-25 genes
gs <- makeExampleData(what="gs", gnames=names(se))

# (3) make artificial enrichment analysis results:
# 2 ea methods with 5 significantly enriched gene sets each
ora.res <- makeExampleData(what="ea.res", method="ora", se=se, gs=gs)
gsea.res <- makeExampleData(what="ea.res", method="gsea", se=se, gs=gs)

# (4) combining the results
```

```

res.list <- list(ora.res, gsea.res)
comb.res <- combResults(res.list)

# (5) result visualization and exploration
gsRanking(comb.res)

# user-defined ranking and combination functions
# (a) dummy ranking, give 1:nrow(res.tbl)
dummy.rank <- function(res.tbl) seq_len(nrow(res.tbl))

# (b) weighted average for combining ranks
wavg <- function(r) mean(c(1,2) * r)

comb.res <- combResults(res.list, rank.fun=dummy.rank, comb.fun=wavg)

```

compileGRN

Compilation of a gene regulatory network from pathway databases

Description

To perform network-based enrichment analysis a gene regulatory network (GRN) is required. There are well-studied processes and organisms for which comprehensive and well-annotated regulatory networks are available, e.g. the RegulonDB for *E. coli* and Yeasttract for *S. cerevisiae*. However, in many cases such a network is missing. A first simple workaround is to compile a network from regulations in pathway databases such as KEGG.

Usage

```

compileGRN(org, db = "kegg", act.inh = TRUE, map2entrez = TRUE,
  keep.type = FALSE, kegg.native = FALSE)

```

Arguments

org	An organism in KEGG three letter code, e.g. 'hsa' for 'Homo sapiens'. Alternatively, and mainly for backward compatibility, this can also be either a list of KEGGPathway objects or an absolute file path of a zip compressed archive of pathway xml files in KGML format.
db	Pathway database. This should be one or more DBs out of 'kegg', 'reactome', 'biocarta', and 'nci'. See pathwayDatabases for available DBs of the respective organism. Default is 'kegg'. Note: when dealing with non-model organisms, GRN compilation is currently only supported directly from KEGG (the argument <code>kegg.native</code> should accordingly be set to TRUE).
act.inh	Should gene regulatory interactions be classified as activating (+) or inhibiting (-)? If TRUE, this will drop interactions for which such a classification cannot be made (e.g. binding events). Otherwise, all interactions found in the pathway DB will be included. Default is TRUE.
map2entrez	Should gene identifiers be mapped to NCBI Entrez Gene IDs? This only applies to Reactome and NCI as they both use UNIPROT IDs. This is typically recommended when using the GRN for network-based enrichment analysis with the <code>EnrichmentBrowser</code> . Default is TRUE.

keep.type	Should the original interaction type descriptions be kept? If TRUE, this will keep the long description of interaction types as found in the original KGML and BioPax pathway files. Default is FALSE.
kegg.native	For KEGG: should the GRN be compiled from the native KGML files or should graphite's pathway topology conversion be used? See the vignette of the graphite package for details. This is mostly for backward compatibility. Default is FALSE. Note: when dealing with non-model organisms (not supported by graphite) this argument should be set to TRUE.

Value

The GRN in plain matrix format. Two columns named FROM (the regulator) and TO (the regulated gene) are guaranteed. Additional columns, named TYPE and LONG.TYPE, are included if option `act.inh` or `keep.type` is activated.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

See Also

[pathwayDatabases](#), [pathways](#), [KEGGPathway](#), [parseKGML](#), [downloadPathways](#)

Examples

```
kegg.grn <- compileGRN(org="hsa", db="kegg")
```

configEBrowser

Configuring the EnrichmentBrowser

Description

Function to get and set configuration parameters determining the default behavior of the EnrichmentBrowser

Usage

```
configEBrowser(key, value = NULL)
```

Arguments

key	Configuration parameter.
value	Value to overwrite the current value of key.

Details

Important colData, rowData, and result column names:

- SMPL.COL: colData column storing the sample IDs (default: "SAMPLE")
- GRP.COL: colData column storing binary group assignment (default: "GROUP")
- BLK.COL: colData column defining paired samples or sample blocks (default: "BLOCK")
- PRB.COL: rowData column storing probe/feature IDs ("PROBEID", read-only)
- EZ.COL: rowData column storing gene ENTREZ IDs ("ENTREZID", read-only)
- SYM.COL: rowData column storing gene symbols ("SYMBOL", read-only)
- GN.COL: rowData column storing gene names ("GENENAME", read-only)
- FC.COL: rowData column storing (log2) fold changes of differential expression between sample groups (default: "FC")
- ADJP.COL: rowData column storing adjusted (corrected for multiple testing) p-values of differential expression between sample groups (default: "ADJ.PVAL")
- GS.COL: result table column storing gene set IDs (default: "GENE.SET")
- PVAL.COL: result table column storing gene set significance (default: "PVAL")
- PMID.COL: gene table column storing PUBMED IDs ("PUBMED", read-only)

Important URLs (all read-only):

- NCBI.URL: <http://www.ncbi.nlm.nih.gov/>
- PUBMED.URL: <http://www.ncbi.nlm.nih.gov/pubmed/>
- GENE.URL: <http://www.ncbi.nlm.nih.gov/gene/>
- KEGG.URL: <http://www.genome.jp/dbget-bin/>
- KEGG.GENE.URL: http://www.genome.jp/dbget-bin/www_bget?
- KEGG.SHOW.URL: http://www.genome.jp/dbget-bin/show_pathway?
- GO.SHOW.URL: <http://amigo.geneontology.org/amigo/term/>

Default output directory:

- EBROWSER.HOME: `rappdirs::user_data_dir("EnrichmentBrowser")`
- OUTDIR.DEFAULT: `file.path(EBROWSER.HOME, "results")`

Gene set size:

- GS.MIN.SIZE: minimum number of genes per gene set (default: 5)
- GS.MAX.SIZE: maximum number of genes per gene set (default: 500)

Result appearance:

- RESULT.TITLE: (default: "Table of Results")
- NR.SHOW: maximum number of entries to show (default: 20)

Value

If `is.null(value)` this returns the value of the selected configuration parameter. Otherwise, it updates the selected parameter with the given value.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

Examples

```
# getting config information
configEBrowser("GS.MIN.SIZE")

# setting config information
# WARNING: this is for advanced users only!
# inappropriate settings will impair EnrichmentBrowser's functionality
configEBrowser(key="GS.MIN.SIZE", value=3)
```

deAna

Differential expression analysis between two sample groups

Description

The function carries out a differential expression analysis between two sample groups. Resulting fold changes and derived p-values are returned. Raw p-values are corrected for multiple testing.

Usage

```
deAna(expr, grp = NULL, blk = NULL, de.method = c("limma", "edgeR",
"DESeq2"), padj.method = "BH", stat.only = FALSE, min.cpm = 2)
```

Arguments

expr	Expression data. A numeric matrix. Rows correspond to genes, columns to samples. Alternatively, this can also be an object of class SummarizedExperiment .
grp	*BINARY* group assignment for the samples. Use '0' and '1' for unaffected (controls) and affected (cases) samples, respectively. If NULL, this is assumed to be defined via a column named 'GROUP' in the colData slot if 'expr' is a SummarizedExperiment .
blk	Optional. For paired samples or sample blocks. This can also be defined via a column named 'BLOCK' in the colData slot if 'expr' is a SummarizedExperiment .
de.method	Differential expression method. Use 'limma' for microarray and RNA-seq data. Alternatively, differential expression for RNA-seq data can be also calculated using edgeR ('edgeR') or DESeq2 ('DESeq2'). Defaults to 'limma'.
padj.method	Method for adjusting p-values to multiple testing. For available methods see the man page of the stats function p.adjust . Defaults to 'BH'.
stat.only	Logical. Should only the test statistic be returned? This is mainly for internal use, in order to carry out permutation tests on the DE statistic for each gene. Defaults to FALSE.
min.cpm	In case of RNA-seq data: should genes not satisfying a minimum counts-per-million (cpm) threshold be excluded from the analysis? This is typically recommended. See the edgeR vignette for details. The default filter is to exclude genes with $\text{cpm} < 2$ in more than half of the samples.

Value

A DE-table with measures of differential expression for each gene/row, i.e. a two-column matrix with log2 fold changes in the 1st column and derived p-values in the 2nd column. If 'expr' is a [SummarizedExperiment](#), the DE-table will be automatically appended to the `rowData` slot.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

See Also

[readSE](#) for reading expression data from file, [normalize](#) for normalization of expression data, [voom](#) for preprocessing of RNA-seq data, [p.adjust](#) for multiple testing correction, [eBayes](#) for DE analysis with limma, [glmFit](#) for DE analysis with edgeR, and [DESeq](#) for DE analysis with DESeq2.

Examples

```
# (1) microarray data: intensity measurements
maSE <- makeExampleData(what="SE", type="ma")
maSE <- deAna(maSE)
rowData(maSE, use.names=TRUE)

# (2) RNA-seq data: read counts
rseqSE <- makeExampleData(what="SE", type="rseq")
rseqSE <- deAna(rseqSE, de.method="DESeq2")
rowData(rseqSE, use.names=TRUE)
```

downloadPathways	<i>Download of KEGG pathways for a particular organism</i>
------------------	--

Description

The function downloads all metabolic and non-metabolic pathways in KEGG XML format for a specified organism.

Usage

```
downloadPathways(org, cache = TRUE, out.dir = NULL, zip = FALSE)
```

Arguments

org	Organism in KEGG three letter code, e.g. 'hsa' for 'homo sapiens'.
cache	Logical. Should a locally cached version used if available? Defaults to TRUE.
out.dir	Output directory. If not null, pathways are written to files in the specified directory.
zip	Logical. In case pathways are written to file ('out.dir' is not null): should output files be zipped?

Value

if(is.null(out.dir)): a list of KEGGPathway objects else: none, as pathways are written to file

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

See Also

[keggList](#), [keggGet](#), [KEGGPathway](#), [parseKGM](#)

Examples

```
pwys <- downloadPathways("hsa")
```

 eaBrowse

Exploration of enrichment analysis results

Description

Functions to extract a flat gene set ranking from an enrichment analysis result object and to detailedly explore it.

Usage

```
eaBrowse(res, nr.show = -1, graph.view = NULL, html.only = FALSE,
  out.dir = NULL, report.name = NULL)
```

```
gsRanking(res, signif.only = TRUE)
```

Arguments

res	Enrichment analysis result list (as returned by the functions sbea and nbea).
nr.show	Number of gene sets to show. As default all statistically significant gene sets are displayed.
graph.view	Optional. Should a graph-based summary (reports and visualizes consistency of regulations) be created for the result? If specified, it needs to be a gene regulatory network, i.e. either an absolute file path to a tabular file or a character matrix with exactly <i>*THREE*</i> cols; 1st col = IDs of regulating genes; 2nd col = corresponding regulated genes; 3rd col = regulation effect; Use '+' and '-' for activation/inhibition.
html.only	Logical. Should the html file only be written (without opening the browser to view the result page)? Defaults to FALSE.
out.dir	Output directory. If NULL, defaults to a timestamp-generated subdirectory of <code>configEBrowser("OUTDIR.DEFAULT")</code> .
report.name	Name of the HTML report. If NULL, defaults to the enrichment method used.
signif.only	Logical. Display only those gene sets in the ranking, which satisfy the significance level? Defaults to TRUE.

Value

gsRanking: [DataFrame](#) with gene sets ranked by the corresponding p-value;

eaBrowse: none, opens the browser to explore results.

If not instructed otherwise (via argument `out.dir`), the main HTML report and associated files are written to `configEBrowser("OUTDIR.DEFAULT")`. See `?configEBrowser` to change the location. If `html.only=FALSE`, the HTML report will automatically be opened in your default browser.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

See Also

[sbea](#), [nbea](#), [combResults](#)

Examples

```
# real data
exprs.file <- system.file("extdata/exprs.tab", package="EnrichmentBrowser")
cdat.file <- system.file("extdata/colData.tab", package="EnrichmentBrowser")
rdat.file <- system.file("extdata/rowData.tab", package="EnrichmentBrowser")
probeSE <- readSE(exprs.file, cdat.file, rdat.file)
geneSE <- probe2gene(probeSE)
geneSE <- deAna(geneSE)
metadata(geneSE)$annotation <- "hsa"

# artificial enrichment analysis results
gs <- makeExampleData(what="gs", gnames=names(geneSE))
ea.res <- makeExampleData(what="ea.res", method="ora", se=geneSE, gs=gs)

# (5) result visualization and exploration
gsRanking(ea.res)

out.dir <- configEBrowser("OUTDIR.DEFAULT")
eaBrowse(ea.res, out.dir=out.dir, report.name="oraReport")
```

 ebrowser

Seamless navigation through enrichment analysis results

Description

This is the all-in-one wrapper function to perform the standard enrichment analysis pipeline implemented in the `EnrichmentBrowser` package.

Usage

```
ebrowser(meth, exprs, cdat, rdat, org, data.type = c(NA, "ma", "rseq"),
  norm.method = "quantile", de.method = "limma", gs, grn = NULL,
  perm = 1000, alpha = 0.05, beta = 1, comb = FALSE,
  browse = TRUE, nr.show = -1, out.dir = NULL,
  report.name = "index", ...)
```

Arguments

meth	Enrichment analysis method(s). See sbeaMethods and nbeaMethods for currently supported enrichment analysis methods. See also sbea and nbea for details.
exprs	Expression matrix. A tab separated text file containing the expression values (microarray: intensity measurements, RNA-seq: read counts). Columns = samples/subjects; rows = features/probes/genes; NO headers, row or column names. Alternatively, this can be a SummarizedExperiment , assuming the expression matrix in the assays slot. See details.
cdat	Column (phenotype) data. A tab separated text file containing annotation information for the samples in either <i>*two or three*</i> columns. NO headers, row or column names. The number of rows/samples in this file should match the number of columns/samples of the expression matrix. The 1st column is reserved for the sample IDs; The 2nd column is reserved for a <i>*BINARY*</i> group assignment. Use '0' and '1' for unaffected (controls) and affected (cases) sample class, respectively. For paired samples or sample blocks a third column is expected that defines the blocks. If 'exprs' is a SummarizedExperiment , the 'cdat' argument can be left unspecified, which then expects group and optional block assignments in respectively named columns 'GROUP' (mandatory) and 'BLOCK' (optional) in the colData slot.
rdat	Row (feature) data. A tab separated text file containing annotation information for the features. In case of probe level data: exactly <i>*TWO*</i> columns; 1st col = probe/feature IDs; 2nd col = corresponding gene ID for each feature ID in 1st col. In case of gene level data: the gene IDs newline-separated (i.e. just <i>*one*</i> column). It is recommended to use <i>*ENTREZ*</i> gene IDs (to benefit from downstream visualization and exploration functionality of the EnrichmentBrowser). NO headers, row or column names. The number of rows (features/probes/genes) in this file should match the number of rows/features of the expression matrix. Alternatively, this can also be the ID of a recognized platform such as 'hgu95av2' (Affymetrix Human Genome U95 chip) or 'ecoli2' (Affymetrix E. coli Genome 2.0 Array). If 'exprs' is a SummarizedExperiment , the 'rdat' argument can be left unspecified, which then expects probe and corresponding Entrez Gene IDs in respectively named columns 'PROBEID' and 'ENTREZID' in the rowData slot.
org	Organism under investigation in KEGG three letter code, e.g. 'hsa' for 'Homo sapiens'. See also kegg.species.code to convert your organism of choice to KEGG three letter code.
data.type	Expression data type. Use 'ma' for microarray and 'rseq' for RNA-seq data. If NA, data.type is automatically guessed. If the expression values in 'exprs' are decimal numbers they are assumed to be microarray intensities. Whole numbers are assumed to be RNA-seq read counts. Defaults to NA.
norm.method	Determines whether and how the expression data should be normalized. For available microarray normalization methods see the man page of the limma function normalizeBetweenArrays . For available RNA-seq normalization methods see the man page of the EDASeq function betweenLaneNormalization . Defaults to 'quantile', i.e. normalization is carried out so that quantiles between arrays/lanes/samples are equal. Use 'none' to indicate that the data is already normalized and should not be normalized by ebrowser. See the man page of normalize for details.

de.method	Determines which method is used for per-gene differential expression analysis. See the man page of <code>deAna</code> for details. Defaults to 'limma', i.e. differential expression is calculated based on the typical limma <code>lmFit</code> procedure.
gs	Gene sets. Either a list of gene sets (character vectors of gene IDs) or a text file in GMT format storing all gene sets under investigation.
grn	Gene regulatory network. Either an absolute file path to a tabular file or a character matrix with exactly *THREE* cols; 1st col = IDs of regulating genes; 2nd col = corresponding regulated genes; 3rd col = regulation effect; Use '+' and '-' for activation/inhibition.
perm	Number of permutations of the sample group assignments. Defaults to 1000. Can also be an integer vector matching the length of 'meth' to assign different numbers of permutations for different methods.
alpha	Statistical significance level. Defaults to 0.05.
beta	Log2 fold change significance level. Defaults to 1 (2-fold).
comb	Logical. Should results be combined if more than one enrichment method is selected? Defaults to FALSE.
browse	Logical. Should results be displayed in the browser for interactive exploration? Defaults to TRUE.
nr.show	Number of gene sets to show. As default all statistical significant gene sets are displayed. Note that this only influences the number of gene sets for which additional visualization will be provided (typically only of interest for the top / significant gene sets). Selected enrichment methods and resulting flat gene set rankings still include the complete number of gene sets under study.
out.dir	Output directory. If NULL, defaults to a timestamp-generated subdirectory of <code>configEBrowser("OUTDIR.DEFAULT")</code> .
report.name	Character. Name of the HTML report. Defaults to "index".
...	Additional arguments passed on to the individual building blocks.

Details

Given flat gene expression data, the data is read in and subsequently subjected to chosen enrichment analysis methods.

The results from different methods can be combined and investigated in detail in the default browser.

On data type and normalization:

Normalization of high-throughput expression data is essential to make results within and between experiments comparable. Microarray (intensity measurements) and RNA-seq (read counts) data exhibit typically distinct features that need to be normalized for. This function wraps commonly used functionality from limma for microarray normalization and from EDASeq for RNA-seq normalization. For specific needs that deviate from standard normalizations, the user should always refer to more specific functions/packages. See also the limma's user guide <http://www.bioconductor.org/packages/limma> for definition and normalization of the different expression data types.

Microarray data is expected to be single-channel. For two-color arrays, it is expected here that normalization within arrays has been already carried out, e.g. using `normalizeWithinArrays` from limma.

RNA-seq data is expected to be raw read counts. Please note that normalization for downstream DE analysis, e.g. with edgeR and DESeq2, is not ultimately necessary (and in some cases even discouraged) as many of these tools implement specific normalization approaches. See the vignette of EDASeq, edgeR, and DESeq2 for details.

Value

None, writes an HTML report and, if selected, opens the browser to explore results. If not instructed otherwise (via argument `out.dir`), the main HTML report and associated files are written to `configEBrowser("OUTDIR.DEFAULT")`. See `?configEBrowser` to change the location. If `browse=TRUE`, the HTML report will automatically be opened in the default browser.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

References

Limma User's guide: <http://www.bioconductor.org/packages/limma>

See Also

[readSE](#) to read expression data from file; [probe2gene](#) to transform probe to gene level expression; [kegg.species.code](#) maps species name to KEGG code. [getGenesets](#) to retrieve gene set databases such as GO or KEGG; [compileGRN](#) to construct a GRN from pathway databases; [sbea](#) to perform set-based enrichment analysis; [nbea](#) to perform network-based enrichment analysis; [combResults](#) to combine results from different methods; [eaBrowse](#) for exploration of resulting gene sets

Examples

```
# expression data from file
exprs.file <- system.file("extdata/exprs.tab", package="EnrichmentBrowser")
cdat.file <- system.file("extdata/colData.tab", package="EnrichmentBrowser")
rdat.file <- system.file("extdata/rowData.tab", package="EnrichmentBrowser")

# getting all human KEGG gene sets
# hsa.gs <- getGenesets(org="hsa", db="kegg")
gs.file <- system.file("extdata/hsa_kegg_gs.gmt", package="EnrichmentBrowser")
hsa.gs <- getGenesets(gs.file)

# output destination
out.dir <- configEBrowser("OUTDIR.DEFAULT")

# set-based enrichment analysis
ebrowser( meth="ora", perm=0,
          exprs=exprs.file, cdat=cdat.file, rdat=rdat.file,
          gs=hsa.gs, org="hsa", nr.show=3,
          out.dir=out.dir, report.name="oraReport")

# compile a gene regulatory network from KEGG pathways
hsa.grn <- compileGRN(org="hsa", db="kegg")

# network-based enrichment analysis
ebrowser( meth="ggea",
          exprs=exprs.file, cdat=cdat.file, rdat=rdat.file,
          gs=hsa.gs, grn=hsa.grn, org="hsa", nr.show=3,
          out.dir=out.dir, report.name="ggeaReport")

# combining results
```

```
ebrowser( meth=c("ora", "ggea"), perm=0, comb=TRUE,
          exprs=exprs.file, cdat=cdat.file, rdat=rdat.file,
          gs=hsa.gs, grn=hsa.grn, org="hsa", nr.show=3,
          out.dir=out.dir, report.name="combReport")
```

getGenesets

Definition of gene sets according to different sources

Description

Functionality for retrieving gene sets for an organism under investigation from databases such as GO and KEGG. Parsing and writing a list of gene sets from/to a flat text file in GMT format is also supported.

The GMT (Gene Matrix Transposed) file format is a tab delimited file format that describes gene sets. In the GMT format, each row represents a gene set. Each gene set is described by a name, a description, and the genes in the gene set. See references.

Usage

```
getGenesets(org, db = c("go", "kegg"), cache = TRUE,
            go.onto = c("BP", "MF", "CC"), go.mode = c("GO.db", "biomart"),
            return.type = c("list", "GeneSetCollection"))
```

```
writeGMT(gs, gmt.file)
```

Arguments

org	An organism in (KEGG) three letter code, e.g. 'hsa' for 'Homo sapiens'. Alternatively, this can also be a text file storing gene sets in GMT format. See details.
db	Database from which gene sets should be retrieved. Currently, either 'go' (default) or 'kegg'.
cache	Logical. Should a locally cached version used if available? Defaults to TRUE.
go.onto	Character. Specifies one of the three GO ontologies: 'BP' (biological process), 'MF' (molecular function), 'CC' (cellular component). Defaults to 'BP'.
go.mode	Character. Determines in which way the gene sets are retrieved. This can be either 'GO.db' or 'biomart'. The 'GO.db' mode creates the gene sets based on BioC annotation packages - which is fast, but represents not necessarily the most up-to-date mapping. In addition, this option is only available for the currently supported model organisms in BioC. The 'biomart' mode downloads the mapping from BioMart - which can be time consuming, but allows to select from a larger range of organisms and contains the latest mappings. Defaults to 'GO.db'.
return.type	Character. Determines whether gene sets are returned as a simple list of gene sets (each being a character vector of gene IDs), or as an object of class GeneSetCollection .
gs	A list of gene sets (character vectors of gene IDs).
gmt.file	Gene set file in GMT format. See details.

Value

For getGenesets: a list of gene sets (vectors of gene IDs). For writeGMT: none, writes to file.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

References

GO: <http://geneontology.org/>

KEGG Organism code http://www.genome.jp/kegg/catalog/org_list.html

GMT file format http://www.broadinstitute.org/cancer/software/gsea/wiki/index.php/Data_formats

See Also

[annFUN](#) for general GO2gene mapping used in the 'GO.db' mode, and the [biomaRt](#) package for general queries to BioMart.

[keggList](#) and [keggLink](#) for accessing the KEGG REST server.

Examples

```
# (1) Typical usage for gene set enrichment analysis with GO:
# Biological process terms based on BioC annotation (for human)
go.gs <- getGenesets(org="hsa", db="go")

# eq.:
# go.gs <- getGenesets(org="hsa", db="go", go.onto="BP", go.mode="GO.db")

# Alternatively:
# downloading from BioMart
# this may take a few minutes ...

go.gs <- getGenesets(org="hsa", db="go", mode="biomart")

# (2) Defining gene sets according to KEGG
kegg.gs <- getGenesets(org="hsa", db="kegg")

# (3) parsing gene sets from GMT
gmt.file <- system.file("extdata/hsa_kegg_gs.gmt", package="EnrichmentBrowser")
gs <- getGenesets(gmt.file)

# (4) writing gene sets to file
writeGMT(gs, gmt.file)
```

 ggeaGraph

GGEA graphs of consistency between regulation and expression

Description

Gene graph enrichment analysis (GGEA) is a network-based enrichment analysis method implemented in the `EnrichmentBrowser` package. The idea of GGEA is to evaluate the consistency of known regulatory interactions with the observed gene expression data. A GGEA graph for a gene set of interest displays the consistency of each interaction in the network that involves a gene set member. Nodes (genes) are colored according to expression (up-/down-regulated) and edges (interactions) are colored according to consistency, i.e. how well the interaction type (activation/inhibition) is reflected in the correlation of the expression of both interaction partners.

Usage

```
ggeaGraph(gs, grn, se, alpha = 0.05, beta = 1, max.edges = 50,
  cons.thresh = 0.7, show.scores = FALSE)

ggeaGraphLegend()
```

Arguments

<code>gs</code>	Gene set under investigation. This should be a character vector of gene IDs.
<code>grn</code>	Gene regulatory network. Character matrix with exactly <i>*THREE*</i> cols; 1st col = IDs of regulating genes; 2nd col = corresponding regulated genes; 3rd col = regulation effect; Use '+' and '-' for activation/inhibition.
<code>se</code>	Expression data given as an object of class <code>SummarizedExperiment</code> .
<code>alpha</code>	Statistical significance level. Defaults to 0.05.
<code>beta</code>	Log2 fold change significance level. Defaults to 1 (2-fold).
<code>max.edges</code>	Maximum number of edges that should be displayed. Defaults to 50.
<code>cons.thresh</code>	Consistency threshold. Graphical parameter that correspondingly increases line width of edges with a consistency above the chosen threshold (defaults to 0.7).
<code>show.scores</code>	Logical. Should consistency scores of the edges be displayed? Defaults to FALSE.

Value

None, plots to a graphics device.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

See Also

[nbea](#) to perform network-based enrichment analysis. [eaBrowse](#) for exploration of resulting gene sets.

Examples

```
# (1) expression data:
# simulated expression values of 100 genes
# in two sample groups of 6 samples each
se <- makeExampleData(what="SE")
se <- deAna(se)

# (2) gene sets:
# draw 10 gene sets with 15-25 genes
gs <- makeExampleData(what="gs", gnames=names(se))

# (3) compiling artificial regulatory network
grn <- makeExampleData(what="grn", nodes=names(se))

# (4) plot consistency graph
ggeaGraph(gs=gs[[1]], grn=grn, se=se)

# (5) get legend
ggeaGraphLegend()
```

idMap

Mapping between gene ID types for the rownames of a Summarized-Experiment

Description

Functionality to map the rownames of a SummarizedExperiment between common gene ID types such as ENSEMBL and ENTREZ.

Usage

```
idMap(se, org = NA, from = "ENSEMBL", to = "ENTREZID",
      multi.to = "first", multi.from = "first")
```

```
idTypes(org)
```

Arguments

se	An object of class SummarizedExperiment . Expects the names to be of gene ID type given in argument from.
org	Character. Organism in KEGG three letter code, e.g. 'hsa' for 'Homo sapiens'. See references.
from	Character. Gene ID type from which should be mapped. Corresponds to the gene ID type of the names of argument se. Note that from is ignored if to is a rowData column of se. Defaults to ENSEMBL.
to	Character. Gene ID type to which should be mapped. Corresponds to the gene ID type the rownames of argument se should be updated with. Note that this can also be the name of a column in the rowData slot of se to specify user-defined mappings in which conflicts have been manually resolved. Defaults to ENTREZID.

- `multi.to` How to resolve 1:many mappings, i.e. multiple to.IDs for a single from.ID? This is passed on to the `multiVals` argument of `mapIds` and can thus take several pre-defined values, but also the form of a user-defined function. However, note that this requires that a single to.ID is returned for each from.ID. Default is "first", which accordingly returns the first to.ID mapped onto the respective from.ID.
- `multi.from` How to resolve many:1 mappings, i.e. multiple from.IDs mapping to the same to.ID? Pre-defined options include:
- 'first' (Default): returns the first from.ID for each to.ID with multiple from.IDs,
 - 'minp': selects the from.ID with minimum p-value (according to the `rowData` column PVAL of `se`),
 - 'maxfc': selects the from.ID with maximum absolute log2 fold change (according to the `rowData` column FC of `se`).
- Note that a user-defined function can also be supplied for custom behaviors. This will be applied for each case where there are multiple from.IDs for a single to.ID, and accordingly takes the arguments `ids` and `se`. The argument `ids` corresponds to the multiple from.IDs from which a single ID should be chosen, e.g. via information available in argument `se`. See examples for a case where `ids` are selected based on a user-defined `rowData` column.

Details

The function 'idTypes' lists the valid values which the arguments 'from' and 'to' can take. This corresponds to the names of the available gene ID types for the mapping.

Value

`idTypes`: character vector listing the available gene ID types for the mapping;

`idMap`: An object of class `SummarizedExperiment`.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

References

KEGG Organism code http://www.genome.jp/kegg/catalog/org_list.html

See Also

`SummarizedExperiment`, `mapIds`, `keytypes`

Examples

```
# create an expression dataset with 3 genes and 3 samples
se <- makeExampleData("SE", nfeat=3, nsmpl=3)
names(se) <- paste0("ENSG000000000", c("003", "005", "419"))
mse <- idMap(se, org="hsa")

# user-defined mapping
rowData(se)$MYID <- c("g1", "g1", "g2")
mse <- idMap(se, to="MYID")
```

```

# data-driven resolving of many:1 mappings

## e.g. select from.ID with lowest p-value
pcol <- configEBrowser("PVAL.COL")
rowData(se)[[pcol]] <- c(0.001, 0.32, 0.15)
mse <- idMap(se, to="MYID", multi.from="minp")

## ... or using a customized function
maxScore <- function(ids, se)
{
  scores <- rowData(se, use.names=TRUE)[ids, "SCORE"]
  ind <- which.max(scores)
  return(ids[ind])
}
rowData(se)$SCORE <- c(125.7, 33.4, 58.6)
mse <- idMap(se, to="MYID", multi.from=maxScore)

```

isAvailable

Is a required package available?

Description

Convenience function for checking and installing required packages.

Usage

```
isAvailable(pkg, type = c("annotation", "software", "data"))
```

Arguments

pkg	Character vector of length 1. A valid name of an existing R package.
type	Character vector of length 1. What type of package is this? Choose one out of 'annotation', 'software', or 'data' package.

Details

Checks whether a required package is available in the library. If yes, the package is loaded via [require](#). If not, the package is optionally installed via [install](#) and, subsequently, loaded via [require](#).

Value

None. See details.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

See Also

[require](#), [install](#)

Examples

```
isAvailable("EnrichmentBrowser", type="software")
```

makeExampleData

Example data for the EnrichmentBrowser package

Description

Functionality to construct example data sets for demonstration. This includes expression data, gene sets, gene regulatory networks, and enrichment analysis results.

Usage

```
makeExampleData(what = c("SE", "gs", "grn", "ea.res"), ...)
```

Arguments

- | | |
|------|--|
| what | Kind of example data set to be constructed. This should be one out of: <ul style="list-style-type: none"> • SE: SummarizedExperiment • gs: Gene set list • grn: Gene regulatory network • ea.res: Enrichment analysis result object as returned by the functions sbea and nbea |
| ... | Additional arguments to fine-tune the specific example data sets. <p>For what='SE':</p> <ul style="list-style-type: none"> • type: Expression data type. Should be either 'ma' (Microarray intensity measurements) or 'rseq' (RNA-seq read counts). • nfeat: Number of features/genes. Defaults to 100. • nsmpl: Number of samples. Defaults to 12. • blk: Create sample blocks. Defaults to TRUE. • norm: Should the expression data be normalized? Defaults to FALSE. • deAna: Should an differential expression analysis be carried out automatically? Defaults to FALSE. <p>For what='gs':</p> <ul style="list-style-type: none"> • gnames: gene names from which the sets will be sampled. Per default the sets will be drawn from c(g1, ..., g100). • n: number of sets. Defaults to 10. • min.size: minimal set size. Defaults to 15. • max.size: maximal set size. Defaults to 25. <p>For what='grn':</p> <ul style="list-style-type: none"> • nodes: gene node names for which edges will be drawn. Per default node names will be c(g1, ..., g100). |

- `edge.node.ratio`: ratio number of edges / number of nodes. Defaults to 3, i.e. creates 3 times more edges than nodes.

For `what='ea.res'`:

- `SE`: SummarizedExperiment. Calls `makeExampleData(what="SE")` per default.
- `gs`: Gene sets. Calls `makeExampleData(what="gs")` per default.
- `method`: Enrichment analysis method. Defaults to 'ora'.
- `alpha`: Statistical significance level. Defaults to 0.05.

Value

Depends on the 'what' argument.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

Examples

```
se <- makeExampleData(what="SE")
```

nbeaMethods

Network-based enrichment analysis (NBEA)

Description

This is the main function for network-based enrichment analysis. It implements and wraps existing implementations of several frequently used methods and allows a flexible inspection of resulting gene set rankings.

Usage

```
nbeaMethods()
```

```
nbea(method = EnrichmentBrowser::nbeaMethods(), se, gs, grn,
      prune.grn = TRUE, alpha = 0.05, perm = 1000,
      padj.method = "none", out.file = NULL, browse = FALSE, ...)
```

Arguments

- | | |
|---------------------|---|
| <code>method</code> | Network-based enrichment analysis method. Currently, the following network-based enrichment analysis methods are supported: 'ggea', 'spia', 'pathnet', 'degraph', 'topologygsa', 'ganpa', 'cepa', and 'netgsa'. Default is 'ggea'. This can also be the name of a user-defined function implementing network-based enrichment. See Details. |
| <code>se</code> | Expression dataset. An object of class SummarizedExperiment . Mandatory minimal annotations: <ul style="list-style-type: none"> • <code>colData</code> column storing binary group assignment (named "GROUP") |

- rowData column storing (log2) fold changes of differential expression between sample groups (named "FC")
- rowData column storing adjusted (corrected for multiple testing) p-values of differential expression between sample groups (named "ADJ.PVAL")

Additional optional annotations:

- colData column defining paired samples or sample blocks (named "BLOCK")
- metadata slot named "annotation" giving the organism under investigation in KEGG three letter code (e.g. "hsa" for Homo sapiens)
- metadata slot named "dataType" indicating the expression data type ("ma" for microarray, "rseq" for RNA-seq)

gs	Gene sets. Either a list of gene sets (character vectors of gene IDs) or a text file in GMT format storing all gene sets under investigation.
grn	Gene regulatory network. Either an absolute file path to a tabular file or a character matrix with exactly <i>*THREE*</i> cols; 1st col = IDs of regulating genes; 2nd col = corresponding regulated genes; 3rd col = regulation effect; Use '+' and '-' for activation/inhibition.
prune.grn	Logical. Should the GRN be pruned? This removes duplicated, self, and reversed edges. Defaults to TRUE.
alpha	Statistical significance level. Defaults to 0.05.
perm	Number of permutations of the expression matrix to estimate the null distribution. Defaults to 1000. If using method='ggea', it is possible to set 'perm=0' to use a fast approximation of gene set significance to avoid permutation testing. See Details.
padj.method	Method for adjusting nominal gene set p-values to multiple testing. For available methods see the man page of the stats function p.adjust . Defaults to 'none', i.e. leaves the nominal gene set p-values unadjusted.
out.file	Optional output file the gene set ranking will be written to.
browse	Logical. Should results be displayed in the browser for interactive exploration? Defaults to FALSE.
...	Additional arguments passed to individual nbea methods. This includes currently: <ul style="list-style-type: none"> • beta: Log2 fold change significance level. Defaults to 1 (2-fold).

For SPIA and NEA:

- sig.stat: decides which statistic is used for determining significant DE genes. Options are:
 - 'p' (Default): genes with p-value below alpha.
 - 'fc': genes with $\text{abs}(\log_2(\text{fold change}))$ above beta
 - '&': p & fc (logical AND)
 - '|': p | fc (logical OR)

For GGEA:

- cons.thresh: edge consistency threshold between -1 and 1. Defaults to 0.2, i.e. only edges of the GRN with consistency ≥ 0.2 are included in the analysis. Evaluation on real datasets has shown that this works best to distinguish relevant gene sets. Use consistency of -1 to include all edges.
- gs.edges: decides which edges of the grn are considered for a gene set under investigation. Should be one out of c('&', '|'), denoting logical AND and OR. respectively. Accordingly, this either includes edges for which regulator AND / OR target gene are members of the investigated gene set.

Details

'ggee': gene graph enrichment analysis, scores gene sets according to consistency within the given gene regulatory network, i.e. checks activating regulations for positive correlation and repressing regulations for negative correlation of regulator and target gene expression (Geistlinger et al., 2011). When using 'ggee' it is possible to estimate the statistical significance of the consistency score of each gene set in two different ways: (1) based on sample permutation as described in the original publication (Geistlinger et al., 2011) or (2) using an approximation in the spirit of Bioconductor's npGSEA package that is much faster.

'spia': signaling pathway impact analysis, combines ORA with the probability that expression changes are propagated across the pathway topology; implemented in Bioconductor's SPIA package (Tarca et al., 2009).

'pathnet': pathway analysis using network information, applies ORA on combined evidence for the observed signal for gene nodes and the signal implied by connected neighbors in the network; implemented in Bioconductor's PathNet package.

'degraph': differential expression testing for gene graphs, multivariate testing of differences in mean incorporating underlying graph structure; implemented in Bioconductor's DEGraph package.

'topologygsa': topology-based gene set analysis, uses Gaussian graphical models to incorporate the dependence structure among genes as implied by pathway topology; implemented in CRAN's topologyGSA package.

'ganpa': gene association network-based pathway analysis, incorporates network-derived gene weights in the enrichment analysis; implemented in CRAN's GANPA package.

'cepa': centrality-based pathway enrichment, incorporates network centralities as node weights mapped from differentially expressed genes in pathways; implemented in CRAN's CePa package.

'netgsa': network-based gene set analysis, incorporates external information about interactions among genes as well as novel interactions learned from data; implemented in CRAN's NetGSA package.

It is also possible to use additional network-based enrichment methods. This requires to implement a function that takes 'se', 'gs', 'grn', 'alpha', and 'perm' as arguments and returns a numeric matrix 'res.tbl' with a mandatory column named 'PVAL' storing the resulting p-value for each gene set in 'gs'. The rows of this matrix must be named accordingly (i.e. rownames(res.tbl) == names(gs)). See examples.

Value

nbeaMethods: a character vector of currently supported methods;

nbea: if(is.null(out.file)): an enrichment analysis result object that can be detailedly explored by calling [eaBrowse](#) and from which a flat gene set ranking can be extracted by calling [gsRanking](#). If 'out.file' is given, the ranking is written to the specified file.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

References

Geistlinger et al. (2011) From sets to graphs: towards a realistic enrichment analysis of transcriptional systems. *Bioinformatics*, 27(13), i366-73.

Tarca et al. (2009) A novel signaling pathway impact analysis. *Bioinformatics*, 25(1):75-82.

See Also

Input: [readSE](#), [probe2gene](#), [getGenesets](#) to retrieve gene set definitions from databases such as GO and KEGG. [compileGRN](#) to construct a GRN from pathway databases.

Output: [gsRanking](#) to rank the list of gene sets. [eaBrowse](#) for exploration of resulting gene sets.

Other: [sbea](#) to perform set-based enrichment analysis. [combResults](#) to combine results from different methods.

Examples

```
# currently supported methods
nbeaMethods()

# (1) expression data:
# simulated expression values of 100 genes
# in two sample groups of 6 samples each
se <- makeExampleData(what="SE")
se <- deAna(se)

# (2) gene sets:
# draw 10 gene sets with 15-25 genes
gs <- makeExampleData(what="gs", gnames=names(se))

# (3) make 2 artificially enriched sets:
sig.genes <- names(se)[rowData(se)$ADJ.PVAL < 0.1]
gs[[1]] <- sample(sig.genes, length(gs[[1]]))
gs[[2]] <- sample(sig.genes, length(gs[[2]]))

# (4) gene regulatory network
grn <- makeExampleData(what="grn", nodes=names(se))

# (5) performing the enrichment analysis
ea.res <- nbea(method="ggea", se=se, gs=gs, grn=grn)

# (6) result visualization and exploration
gsRanking(ea.res, signif.only=FALSE)

# using your own tailored function as enrichment method
dummyNBEA <- function(se, gs, grn, alpha, perm)
{
  sig.ps <- sample(seq(0,0.05, length=1000),5)
  insig.ps <- sample(seq(0.1,1, length=1000), length(gs)-5)
  ps <- sample(c(sig.ps, insig.ps), length(gs))
  score <- sample(1:100, length(gs), replace=TRUE)
  res.tbl <- cbind(score, ps)
  colnames(res.tbl) <- c("SCORE", "PVAL")
  rownames(res.tbl) <- names(gs)
  return(res.tbl[order(ps),])
}

ea.res2 <- nbea(method=dummyNBEA, se=se, gs=gs, grn=grn)
gsRanking(ea.res2)
```

`normalize`*Normalization of microarray and RNA-seq expression data*

Description

This function wraps commonly used functionality from `limma` for microarray normalization and from `EDASeq` for RNA-seq normalization.

Usage

```
normalize(se, norm.method = "quantile", within = FALSE,  
         data.type = c(NA, "ma", "rseq"))
```

Arguments

<code>se</code>	An object of class <code>SummarizedExperiment</code> .
<code>norm.method</code>	Determines how the expression data should be normalized. For available microarray normalization methods see the man page of the <code>limma</code> function <code>normalizeBetweenArrays</code> . For available RNA-seq normalization methods see the man page of the <code>EDASeq</code> function <code>betweenLaneNormalization</code> . Defaults to 'quantile', i.e. normalization is carried out so that quantiles between arrays/lanes/samples are equal. See details.
<code>within</code>	Logical. Is only taken into account if <code>data.type='rseq'</code> . Determine whether GC content normalization should be carried out (as implemented in the <code>EDASeq</code> function <code>withinLaneNormalization</code>). Defaults to <code>FALSE</code> . See details.
<code>data.type</code>	Expression data type. Use 'ma' for microarray and 'rseq' for RNA-seq data. If <code>NA</code> , <code>data.type</code> is automatically guessed. If the expression values in 'se' are decimal numbers they are assumed to be microarray intensities. Whole numbers are assumed to be RNA-seq read counts. Defaults to <code>NA</code> .

Details

Normalization of high-throughput expression data is essential to make results within and between experiments comparable. Microarray (intensity measurements) and RNA-seq (read counts) data exhibit typically distinct features that need to be normalized for. For specific needs that deviate from standard normalizations, the user should always refer to more specific functions/packages. See also the `limma`'s user guide <http://www.bioconductor.org/packages/limma> for definition and normalization of the different expression data types.

Microarray data is expected to be single-channel. For two-color arrays, it is expected here that normalization within arrays has been already carried out, e.g. using `normalizeWithinArrays` from `limma`.

RNA-seq data is expected to be raw read counts. Please note that normalization for downstream DE analysis, e.g. with `edgeR` and `DESeq2`, is not ultimately necessary (and in some cases even discouraged) as many of these tools implement specific normalization approaches. See the vignette of `EDASeq`, `edgeR`, and `DESeq2` for details.

Value

An object of class `SummarizedExperiment`.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

See Also

[readSE](#) for reading expression data from file;
[normalizeWithinArrays](#) and [normalizeBetweenArrays](#) for normalization of microarray data;
[withinLaneNormalization](#) and [betweenLaneNormalization](#) for normalization of RNA-seq data.

Examples

```
#
# (1) simulating expression data: 100 genes, 12 samples
#

# (a) microarray data: intensity measurements
maSE <- makeExampleData(what="SE", type="ma")

# (b) RNA-seq data: read counts
rseqSE <- makeExampleData(what="SE", type="rseq")

#
# (2) Normalization
#

# (a) microarray ...
normSE <- normalize(maSE)

# (b) RNA-seq ...
normSE <- normalize(rseqSE)

# ... normalize also for GC content
gc.content <- rnorm(100, 0.5, sd=0.1)
rowData(rseqSE)$gc <- gc.content

normSE <- normalize(rseqSE, within=TRUE)
```

plots

Visualization of gene expression

Description

Visualization of differential gene expression via heatmap, p-value histogram and volcano plot (fold change vs. p-value).

Usage

`pdistr(p)`

`volcano(fc, p)`

`exprsHeatmap(expr, grp, scale.rows = TRUE, log.thresh = 100)`

Arguments

p	Numeric vector of p-values for each gene.
fc	Numeric vector of fold changes (typically on log2 scale).
expr	Expression matrix. Rows correspond to genes, columns to samples.
grp	<i>*BINARY*</i> group assignment for the samples. Use '0' and '1' for unaffected (controls) and affected (cases) samples, respectively.
scale.rows	Should rows of the expression matrix be scaled for better visibility of expression differences between sample groups? Defaults to TRUE.
log.thresh	Threshold for log2-transformation of the expression matrix. Particularly useful for heatmap visualization of RNA-seq read count data, where the max and the min of the expression matrix typically differ by several orders of magnitude. If the difference between min and max of the expression matrix is greater than the indicated threshold, log2-transformation is applied.

Value

None, plots to a graphics device.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

See Also

[deAna](#) for differential expression analysis, [heatmap](#) and [truehist](#) for generic plotting.

Examples

```
# (1) simulating expression data: 100 genes, 12 samples
se <- makeExampleData(what="SE")

# plot heatmap
exprsHeatmap(expr=assay(se), grp=as.factor(se$GROUP))

# (2) DE analysis
se <- deAna(se)
pdistr(rowData(se)$ADJ.PVAL)
volcano(fc=rowData(se)$FC, p=rowData(se)$ADJ.PVAL)
```

probe2gene

Transformation of probe level expression to gene level expression

Description

Transforms expression data on probe level to gene level expression by summarizing all probes that are annotated to a particular gene.

Usage

```
probe2gene(probeSE, chip = NA, from = "PROBEID", to = "ENTREZID",
  multi.to = "first", multi.from = "mean")
```

Arguments

probeSE	Probe expression data. An object of class <code>SummarizedExperiment</code> . Make sure that the <code>metadata</code> contains an element named <code>annotation</code> that provides the corresponding ID of a recognized platform such as hgu95av2 (Affymetrix Human Genome U95 chip). This requires that a corresponding <code>.db</code> package exists (see http://www.bioconductor.org/packages/release/BiocViews.html#___ChipName for available chips/packages) and that you have it installed. Alternatively, the mapping from probe to gene can also be defined in the <code>rowData</code> slot via two columns named (i) PROBEID for the platform-specific probe ID, and (ii) ENTREZID for the corresponding NCBI Entrez Gene ID.
chip	Character. The ID of a recognized microarray platform. Only required if not provided in the <code>metadata</code> of <code>probeSE</code> via an element named <code>annotation</code> .
from	Character. ID type from which should be mapped. Corresponds to the ID type of the names of argument <code>se</code> , with the default PROBEID being appropriate if the mapping is based on Bioconductor annotation packages. Note that <code>from</code> is ignored if <code>to</code> is a <code>rowData</code> column of <code>probeSE</code> .
to	Character. Gene ID type to which should be mapped. Corresponds to the gene ID type the rownames of argument <code>probeSE</code> should be updated with. Note that this can also be the name of a column in the <code>rowData</code> slot of <code>probeSE</code> to specify user-defined mappings in which conflicts have been manually resolved. Defaults to ENTREZID.
multi.to	How to resolve 1:many mappings, i.e. multiple gene IDs for a single probe ID? This is passed on to the <code>multiVals</code> argument of <code>mapIds</code> and can thus take several pre-defined values, but also the form of a user-defined function. However, note that this requires that a single gene ID is returned for each probe ID. Default is "first", which accordingly returns the first gene ID mapped onto the respective probe ID.
multi.from	How to resolve many:1 mappings, i.e. multiple probe IDs mapping to the same gene ID? Pre-defined options include: <ul style="list-style-type: none"> 'mean' (Default): updates the respective gene expression with the average over the expression of all probes mapping to that gene, 'first': returns the first probe ID for each gene ID with multiple probe IDs, 'minp' selects the probe ID with minimum p-value (according to the <code>rowData</code> column PVAL of <code>probeSE</code>), 'maxfc' selects the probe ID with maximum absolute log2 fold change (according to the <code>rowData</code> column FC of <code>probeSE</code>).

Value

A `SummarizedExperiment` on gene level.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

See Also

[readSE](#) for reading expression data from file, [deAna](#) for differential expression analysis.

Examples

```
# (1) reading the expression data from file
exprs.file <- system.file("extdata/exprs.tab", package="EnrichmentBrowser")
cdat.file <- system.file("extdata/colData.tab", package="EnrichmentBrowser")
rdat.file <- system.file("extdata/rowData.tab", package="EnrichmentBrowser")
probeSE <- readSE(exprs.file, cdat.file, rdat.file)
geneSE <- probe2gene(probeSE)
```

readSE	<i>Reading gene expression data from file</i>
--------	---

Description

The function reads in plain expression data from file with minimum annotation requirements for the colData and rowData slots.

Usage

```
readSE(assay.file, cdat.file, rdat.file, data.type = c(NA, "ma", "rseq"),
       NA.method = c("mean", "rm", "keep"))
```

Arguments

assay.file	Expression matrix. A tab separated text file containing expression values. Columns = samples/subjects; rows = features/probes/genes; NO headers, row or column names.
cdat.file	Column (phenotype) data. A tab separated text file containing annotation information for the samples in either <i>*two or three*</i> columns. NO headers, row or column names. The number of rows/samples in this file should match the number of columns/samples of the expression matrix. The 1st column is reserved for the sample IDs; The 2nd column is reserved for a <i>*BINARY*</i> group assignment. Use '0' and '1' for unaffected (controls) and affected (cases) sample class, respectively. For paired samples or sample blocks a third column is expected that defines the blocks.
rdat.file	Row (feature) data. A tab separated text file containing annotation information for the features. In case of probe level data: exactly <i>*TWO*</i> columns; 1st col = probe/feature IDs; 2nd col = corresponding gene ID for each feature ID in 1st col. In case of gene level data: the gene IDs newline-separated (i.e. just <i>*one*</i> column). It is recommended to use <i>*ENTREZ*</i> gene IDs (to benefit from downstream visualization and exploration functionality of the EnrichmentBrowser). NO headers, row or column names. The number of rows (features/probes/genes) in this file should match the number of rows/features of the expression matrix. Alternatively, this can also be the ID of a recognized platform such as 'hgu95av2' (Affymetrix Human Genome U95 chip) or 'ecoli2' (Affymetrix E. coli Genome 2.0 Array).

data.type	Expression data type. Use 'ma' for microarray and 'rseq' for RNA-seq data. If NA, data.type is automatically guessed. If the expression values in the expression matrix are decimal numbers, they are assumed to be microarray intensities. Whole numbers are assumed to be RNA-seq read counts. Defaults to NA.
NA.method	Determines how to deal with NA's (missing values). This can be one out of: <ul style="list-style-type: none"> • mean: replace NA by the mean over all samples for one feature at a time. removed. • keep: do nothing. Missing values are kept (which, however, can then cause several issues in the downstream analysis) Defaults to 'mean'.

Value

An object of class [SummarizedExperiment](#).

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

See Also

[SummarizedExperiment](#)

Examples

```
# reading the expression data from file
assay.file <- system.file("extdata/exprs.tab", package="EnrichmentBrowser")
cdat.file <- system.file("extdata/colData.tab", package="EnrichmentBrowser")
rdat.file <- system.file("extdata/rowData.tab", package="EnrichmentBrowser")
se <- readSE(assay.file, cdat.file, rdat.file)
```

sbeaMethods

Set-based enrichment analysis (SBEA)

Description

This is the main function for the enrichment analysis of gene sets. It implements and wraps existing implementations of several frequently used methods and allows a flexible inspection of resulting gene set rankings.

Usage

```
sbeaMethods()
```

```
sbea(method = EnrichmentBrowser::sbeaMethods(), se, gs, alpha = 0.05,
      perm = 1000, padj.method = "none", out.file = NULL,
      browse = FALSE, ...)
```

Arguments

method	Set-based enrichment analysis method. Currently, the following set-based enrichment analysis methods are supported: 'ora', 'safe', 'gsea', 'padog', 'roast', 'camera', 'gsa', 'gsva', 'globaltest', 'samgs', 'ebm', and 'mgsa'. For basic ora also set 'perm=0'. Default is 'ora'. This can also be the name of a user-defined function implementing set-based enrichment. See Details.
se	Expression dataset. An object of class <code>SummarizedExperiment</code> . Mandatory minimal annotations: <ul style="list-style-type: none"> • colData column storing binary group assignment (named "GROUP") • rowData column storing (log2) fold changes of differential expression between sample groups (named "FC") • rowData column storing adjusted (corrected for multiple testing) p-values of differential expression between sample groups (named "ADJ.PVAL") Additional optional annotations: <ul style="list-style-type: none"> • colData column defining paired samples or sample blocks (named "BLOCK") • metadata slot named "annotation" giving the organism under investigation in KEGG three letter code (e.g. "hsa" for Homo sapiens) • metadata slot named "dataType" indicating the expression data type ("ma" for microarray, "rseq" for RNA-seq)
gs	Gene sets. Either a list of gene sets (character vectors of gene IDs) or a text file in GMT format storing all gene sets under investigation.
alpha	Statistical significance level. Defaults to 0.05.
perm	Number of permutations of the sample group assignments. Defaults to 1000. For basic ora set 'perm=0'. Using method="gsea" and 'perm=0' invokes the permutation approximation from the npGSEA package.
padj.method	Method for adjusting nominal gene set p-values to multiple testing. For available methods see the man page of the stats function <code>p.adjust</code> . Defaults to 'none', i.e. leaves the nominal gene set p-values unadjusted.
out.file	Optional output file the gene set ranking will be written to.
browse	Logical. Should results be displayed in the browser for interactive exploration? Defaults to FALSE.
...	Additional arguments passed to individual sbea methods. This includes currently for ORA and MGSA: <ul style="list-style-type: none"> • beta: Log2 fold change significance level. Defaults to 1 (2-fold). • sig.stat: decides which statistic is used for determining significant DE genes. Options are: <ul style="list-style-type: none"> – 'p' (Default): genes with p-value below alpha. – 'fc': genes with $\text{abs}(\log_2(\text{fold change}))$ above beta – '&': p & fc (logical AND) – 'l': p fc (logical OR)

Details

'ora': overrepresentation analysis, simple and frequently used test based on the hypergeometric distribution (see Goeman and Buhlmann, 2007, for a critical review).

'safe': significance analysis of function and expression, generalization of ORA, includes other test statistics, e.g. Wilcoxon's rank sum, and allows to estimate the significance of gene sets by sample permutation; implemented in the safe package (Barry et al., 2005).

'gsea': gene set enrichment analysis, frequently used and widely accepted, uses a Kolmogorov-Smirnov statistic to test whether the ranks of the p-values of genes in a gene set resemble a uniform distribution (Subramanian et al., 2005).

'padog': pathway analysis with down-weighting of overlapping genes, incorporates gene weights to favor genes appearing in few pathways versus genes that appear in many pathways; implemented in the PADOG package.

'roast': rotation gene set test, uses rotation instead of permutation for assessment of gene set significance; implemented in the limma and edgeR packages for microarray and RNA-seq data, respectively.

'camera': correlation adjusted mean rank gene set test, accounts for inter-gene correlations as implemented in the limma and edgeR packages for microarray and RNA-seq data, respectively.

'gsa': gene set analysis, differs from GSEA by using the maxmean statistic, i.e. the mean of the positive or negative part of gene scores in the gene set; implemented in the GSA package.

'gsva': gene set variation analysis, transforms the data from a gene by sample matrix to a gene set by sample matrix, thereby allowing the evaluation of gene set enrichment for each sample; implemented in the GSVA package.

'globaltest': global testing of groups of genes, general test of groups of genes for association with a response variable; implemented in the globaltest package.

'samgs': significance analysis of microarrays on gene sets, extends the SAM method for single genes to gene set analysis (Dinu et al., 2007).

'ebm': empirical Brown's method, combines p-values of genes in a gene set using Brown's method to combine p-values from dependent tests; implemented in the EmpiricalBrownsMethod package.

'mgsa': model-based gene set analysis, Bayesian modeling approach taking set overlap into account by working on all sets simultaneously, thereby reducing the number of redundant sets; implemented in the mgsa package.

It is also possible to use additional set-based enrichment methods. This requires to implement a function that takes 'se', 'gs', 'alpha', and 'perm' as arguments and returns a numeric vector 'ps' storing the resulting p-value for each gene set in 'gs'. This vector must be named accordingly (i.e. names(ps) == names(gs)). See examples.

Value

sbeaMethods: a character vector of currently supported methods;

sbea: if(is.null(out.file)): an enrichment analysis result object that can be detailedly explored by calling [eaBrowse](#) and from which a flat gene set ranking can be extracted by calling [gsRanking](#). If 'out.file' is given, the ranking is written to the specified file.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

References

Goeman and Buhlmann (2007) Analyzing gene expression data in terms of gene sets: methodological issues. *Bioinformatics*, 23, 980-7.

Barry et al. (2005) Significance Analysis of Function and Expression. *Bioinformatics*, 21:1943-9.

Subramanian et al. (2005) Gene Set Enrichment Analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc Natl Acad Sci USA*, 102:15545-50.

Dinu et al. (2007) Improving gene set analysis of microarray data by SAM-GS. *BMC Bioinformatics*, 8:242

See Also

Input: [readSE](#), [probe2gene](#) [getGenesets](#) to retrieve gene sets from databases such as GO and KEGG.

Output: [gsRanking](#) to retrieve the ranked list of gene sets. [eaBrowse](#) for exploration of resulting gene sets.

Other: [nbea](#) to perform network-based enrichment analysis. [combResults](#) to combine results from different methods.

Examples

```
# currently supported methods
sbeaMethods()

# (1) expression data:
# simulated expression values of 100 genes
# in two sample groups of 6 samples each
se <- makeExampleData(what="SE")
se <- deAna(se)

# (2) gene sets:
# draw 10 gene sets with 15-25 genes
gs <- makeExampleData(what="gs", gnames=names(se))

# (3) make 2 artificially enriched sets:
sig.genes <- names(se)[rowData(se)$ADJ.PVAL < 0.1]
gs[[1]] <- sample(sig.genes, length(gs[[1]]))
gs[[2]] <- sample(sig.genes, length(gs[[2]]))

# (4) performing the enrichment analysis
ea.res <- sbea(method="ora", se=se, gs=gs, perm=0)

# (5) result visualization and exploration
gsRanking(ea.res)

# using your own tailored function as enrichment method
dummySBEA <- function(se, gs, alpha, perm)
{
  sig.ps <- sample(seq(0, 0.05, length=1000), 5)
  nsig.ps <- sample(seq(0.1, 1, length=1000), length(gs)-5)
  ps <- sample(c(sig.ps, nsig.ps), length(gs))
  names(ps) <- names(gs)
  return(ps)
}

ea.res2 <- sbea(method=dummySBEA, se=se, gs=gs)
gsRanking(ea.res2)
```

Index

- annFUN, [15](#)
- assays, [11](#)

- betweenLaneNormalization, [11](#), [25](#), [26](#)

- colData, [7](#), [11](#)
- comb.ea.results (combResults), [2](#)
- combResults, [2](#), [10](#), [13](#), [24](#), [33](#)
- compile.grn.from.kegg (compileGRN), [4](#)
- compileGRN, [4](#), [13](#), [24](#)
- config.ebrowser (configEBrowser), [5](#)
- configEBrowser, [5](#)

- DataFrame, [10](#)
- de.ana (deAna), [7](#)
- deAna, [7](#), [12](#), [27](#), [29](#)
- DESeq, [8](#)
- download.kegg.pathways
 (downloadPathways), [8](#)
- downloadPathways, [5](#), [8](#)

- ea.browse (eaBrowse), [9](#)
- eaBrowse, [3](#), [9](#), [13](#), [16](#), [23](#), [24](#), [32](#), [33](#)
- eBayes, [8](#)
- ebrowser, [10](#)
- exprsHeatmap (plots), [26](#)

- GeneSetCollection, [14](#)
- get.go.genesets (getGenesets), [14](#)
- get.kegg.genesets (getGenesets), [14](#)
- getGenesets, [13](#), [14](#), [24](#), [33](#)
- ggea (nbeaMethods), [21](#)
- ggea.graph (ggeaGraph), [16](#)
- ggeaGraph, [16](#)
- ggeaGraphLegend (ggeaGraph), [16](#)
- glmFit, [8](#)
- gs.ranking (eaBrowse), [9](#)
- gsea (sbeaMethods), [30](#)
- gsRanking, [3](#), [23](#), [24](#), [32](#), [33](#)
- gsRanking (eaBrowse), [9](#)

- heatmap, [27](#)

- idMap, [17](#)
- idTypes (idMap), [17](#)

- install, [19](#)
- isAvailable, [19](#)

- kegg.species.code, [11](#), [13](#)
- keggGet, [9](#)
- keggLink, [15](#)
- keggList, [9](#), [15](#)
- KEGGPathway, [4](#), [5](#), [9](#)
- keytypes, [18](#)

- lmFit, [12](#)

- makeExampleData, [20](#)
- map.ids (idMap), [17](#)
- mapIds, [18](#), [28](#)
- metadata, [28](#)

- nbea, [2](#), [3](#), [9–11](#), [13](#), [16](#), [20](#), [33](#)
- nbea (nbeaMethods), [21](#)
- nbeaMethods, [11](#), [21](#)
- normalize, [8](#), [11](#), [25](#)
- normalizeBetweenArrays, [11](#), [25](#), [26](#)
- normalizeWithinArrays, [12](#), [25](#), [26](#)

- ora (sbeaMethods), [30](#)

- p.adjust, [7](#), [8](#), [22](#), [31](#)
- parse.genesets.from.GMT (getGenesets),
 [14](#)
- parseKGM, [5](#), [9](#)
- pathwayDatabases, [4](#), [5](#)
- pathways, [5](#)
- pdistr (plots), [26](#)
- plots, [26](#)
- probe.2.gene.eset (probe2gene), [27](#)
- probe2gene, [13](#), [24](#), [27](#), [33](#)

- read.eset (readSE), [29](#)
- readSE, [8](#), [13](#), [24](#), [26](#), [29](#), [29](#), [33](#)
- require, [19](#)
- rowData, [8](#), [11](#), [17](#), [18](#), [28](#)

- sbea, [2](#), [3](#), [9–11](#), [13](#), [20](#), [24](#)
- sbea (sbeaMethods), [30](#)
- sbeaMethods, [11](#), [30](#)

`spia` (`nbeaMethods`), [21](#)
`SummarizedExperiment`, [7](#), [8](#), [11](#), [16–18](#), [21](#),
[25](#), [28](#), [30](#), [31](#)

`truehist`, [27](#)

`volcano` (plots), [26](#)
`voom`, [8](#)

`withinLaneNormalization`, [25](#), [26](#)
`writeGMT` (`getGenesets`), [14](#)